Unity Surface Cache GI - Getting Started

We're thrilled to give you a first look at **Surface Cache GI**, our first step towards delivering robust, fully dynamic Global Illumination in Unity! This feature preview is intended for you, our most curious creators, to try out and help us shape its future.

Your insights are crucial to making Surface Cache GI the best it can be. Experiment with the feature, test it in your projects, and let us know what you think.

In particular, we're keen to learn about a variety of use cases, so that we can evaluate what type of content works well. Stylized to photoreal, construction/destruction, animated scenes (doors opening/closing), time of day, content running on various platforms.

When you are ready to provide feedback, please take a few moments to fill out our survey. For additional questions or feedback feel free to connect: steven.kent@unity3d.com (product contact).

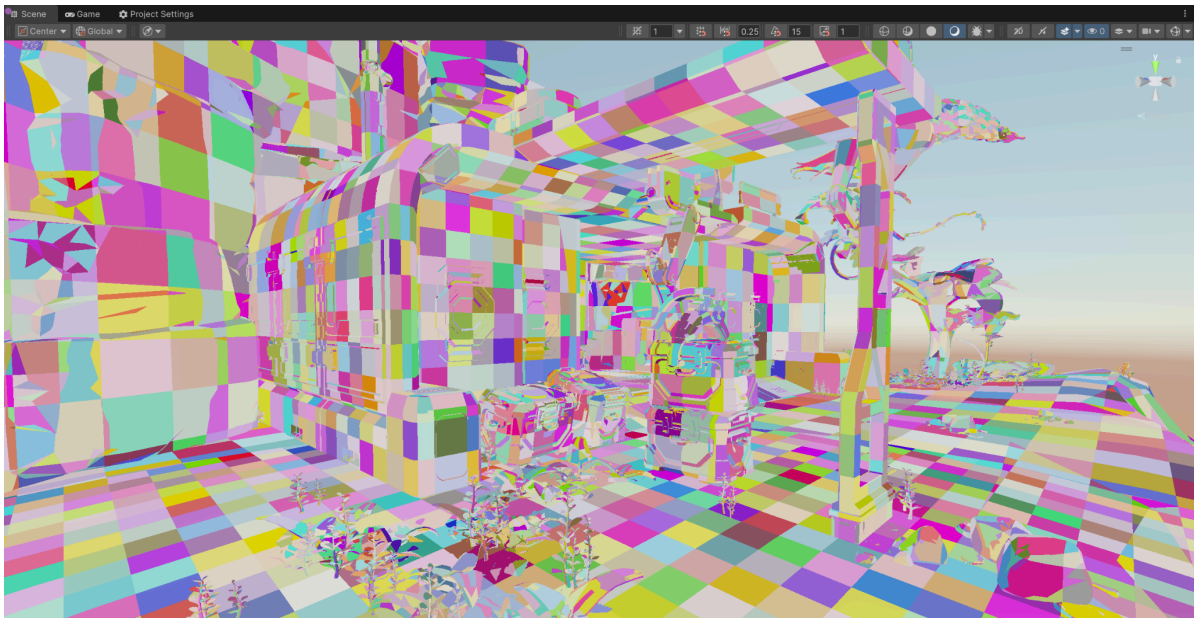
Thank you for being an early adopter and helping us shape the future of dynamic GI in Unity!

What is Surface Cache GI?

Surface Cache GI is a new URP feature that provides real-time indirect diffuse lighting and runs across a broad range of devices. It's designed to make authoring faster, eliminate precomputation, and support scenarios involving dynamic light and geometry such as runtime construction/destruction, procedurally generated content, time of day, and much more. Forget complex workflows and long bake times – Surface Cache GI aims for lighting that "just works."

How it Works

Instead of baking lightmaps or probes, Surface Cache computes diffuse GI in realtime. It works by discretizing the 5D space of all positions (3D) and directions (2D) into a finite number of "patches", mapping surface position and normal direction pairs within its volume into a unique patch index. The size of these patches changes with camera distance so that we have more patches nearer the camera and fewer far away from the camera.



Visualization of scene surface discretized into patches.

During rendering we identify which patches are relevant for the current frame and ensure they are allocated. For each allocated patch, Surface Cache calculates average incoming hemispherical lighting by shooting rays into the scene. If available, hardware accelerated ray tracing is leveraged. Otherwise, a compute-based software ray tracing backend is used instead.

Indirect lighting calculated for each patch using ray tracing.

Finally, we reconstruct per-pixel irradiance by filtering the patch data. This filtering process is done in such a way that we can apply the per-pixel shaded normal to capture fine detail from normal maps.

Indirect lighting reconstructed per pixel based on patch data.

Enabling Surface Cache GI

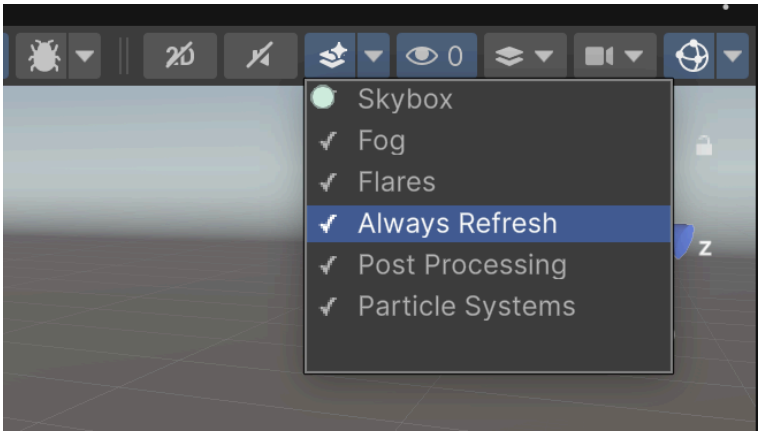
Setup Project

1. Install the latest Alpha version through the Unity Alpha program, at least 6000.6.0a5.
2. Create or load a URP project.

Configure Project

1. Navigate to Project Settings > Player > Other Settings.
2. Under Scripting Define Symbols, add SURFACE_CACHE as a new entry. This will make the Surface Cache feature appear in URP's list of Renderer Features.
3. In Project Settings > Player > Other Settings, disable "Static Batching". Surface Cache currently does not work with "Static Batching" enabled.

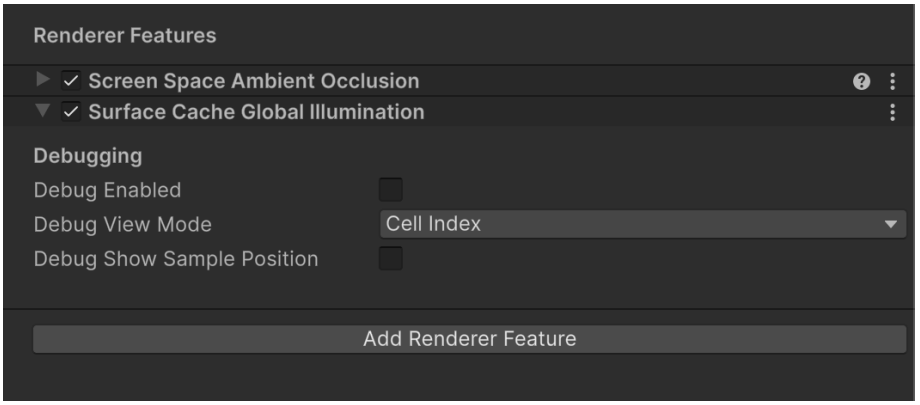
4. Enable “Always Refresh” in the effects dropdown. This is not strictly required but it will make Surface Cache update faster in Scene View.



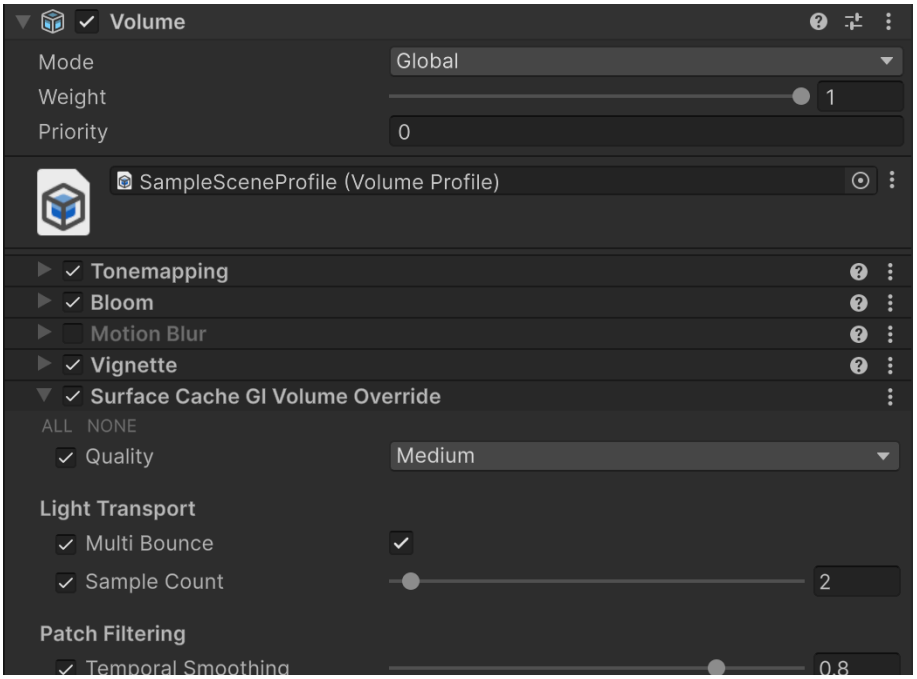
5. If you want to use Surface Cache in a built player, you must set Read/Write=True in the Mesh Importer of all meshes in that build. This step won’t be required in the final version of Surface Cache. See “Known Issues” section for more details.

Add Renderer Feature

1. In your Universal Rendering Data asset, add the Surface Cache Global Illumination Renderer Feature. This enables the feature for the given URP Renderer.

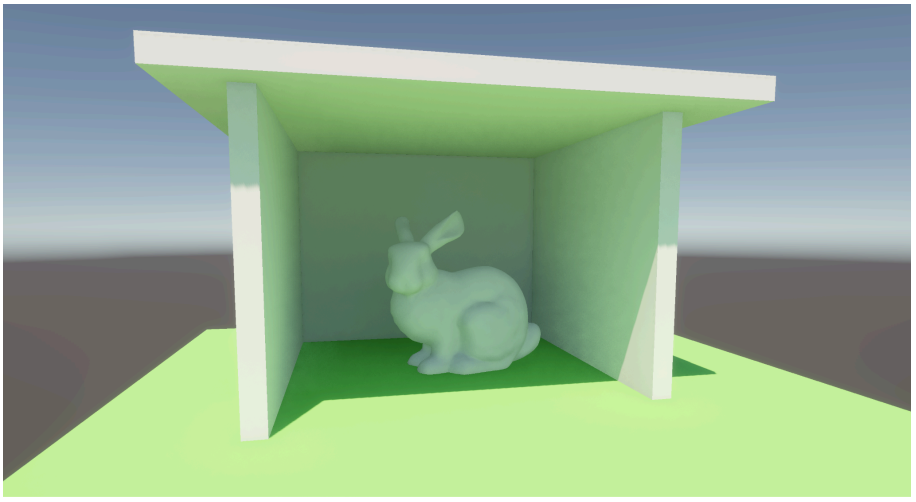


2. In a Volume Component, add a Surface Cache GI Volume override. This is where you configure the behaviour and quality of Surface Cache



That's it!

You should now see realtime indirect lighting in your scene.



Visualization Tools

Before you start configuring Surface Cache, you should first understand how you can visualize its internal state.

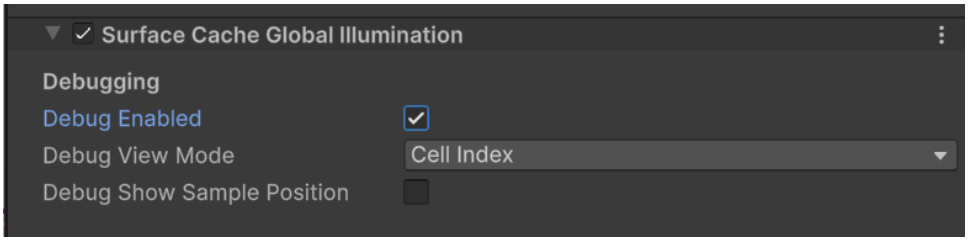
Lighting Debug Mode

Surface Cache provides indirect lighting, also sometimes referred to as Global Illumination (like lightmaps and probes). URP's Rendering Debugger offers a general way to visualize Global Illumination. Navigate to Window → Analysis → Rendering Debugger → Lighting and set Lighting Debug Modes = Global Illumination. If Surface Cache is enabled, this mode shows its output in isolation (e.g. without direct lighting and other effects).

You can inspect Surface Cache output via Rendering Debugger's Global Illumination Lighting Debug Mode.

Surface Cache Debug Mode

Surface Cache itself has debug mode. You can enable this in the Surface Cache Global Illumination Renderer Feature.



This mode can be configured in the following ways:

- **Debug Enabled**: Whether Debug mode is enabled or not.
- **Debug View Mode:**
    - Cell Index: Visualize the discretization of the scene surfaces into patches.

    - Stable Irradiance: Visualizes the filtered lighting for each patch.
    - Fast Irradiance: Visualizes lighting without applying the post temporal filter.
    - Coefficient Of Variation: A statistics used to guide the temporal filter.
    - Drift: A statistics used to guide the temporal filter.
    - Std Dev: A statistics used to guide the temporal filter.
    - Update Count: The number of samples obtained by this patch.
    - Flat Normal: The flat geometric normals which are used to spawn patches.
- **Debug Show Sample Position**: Enabling this will visualize the ray origin used every frame for each patch.

When Debug Mode is enabled the state of the internal ring buffer is visualized at the bottom of the screen. Each color has a specific meaning:

    - Grey/white: Slot is available for use.
    - Green: Slot is in use by a patch spawned by the camera.
    - Yellow: Slot is in use by a patch spawned by another patch.
    - Red: Slot is inactive and is awaiting garbage collection.

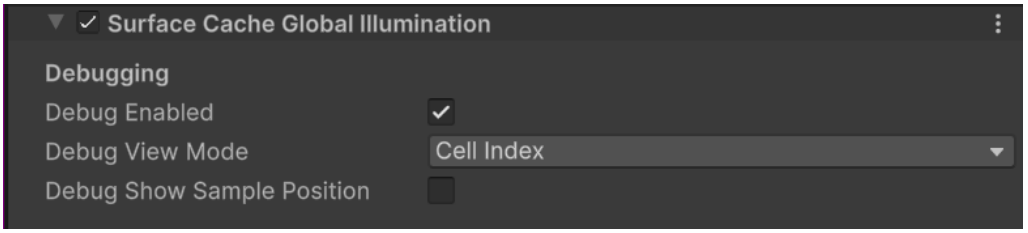
Configuration

In this section we will discuss the settings of the Surface Cache GI Volume Override. Typically, you want to apply these settings to a global volume but it is also possible to have the settings interpolated spatially via several local volumes.

Volume

Surface Cache uses a spatial cascaded structure which discretizes the scene surface into so-called “patches”. These settings control the size, resolution, and distribution of this structure.

Tip: It is useful to visualize the discretization of the structure while you are configuring it. You can do this by setting Debug Enabled = True and Debug View Mode = Cell Index inside the Surface Cache Renderer Feature. Additionally, you may go to Window → Analysis → Rendering Debugger → Lighting and set Lighting Debug Modes = Global Illumination to make this even clearer.



It can be helpful to enable debug mode while configuring the volume.

The volume is controlled via these settings:

- **Size:** The radius of the sphere that Surface Cache must cover. Beyond this distance, no patches will be spawned. Typically you want this size as small as possible, yet such that it still covers your scene.
- **Resolution:** Controls the number of patches inside each cascade. Increasing this value will make each patch smaller but since it doesn't affect the size of the structure, this also means you will get more patches. Often, you want patches near the camera to have an area of between 0,1×0,1 and 0,25×0,25 square units.
- **Cascade Count:** Controls how patches are distributed inside the volume. A single cascade yields uniformly sized patches across the entire volume. However often you want higher resolution nearer the camera. You can achieve this by adding more cascades. The more cascades you add, the higher the resolution will become closer to the camera. Note though that the drawback of adding cascades is that you usually also increase the total number of patches which will hurt performance.
- **Cascade Movement**: When enabled, Surface Cache will use the camera position as the center of the cascade structure. Effectively, this means more smaller patches near the camera and fewer larger patches farther away from the camera. When disabled, Surface Cache will not "follow" the camera and instead use whatever center position was last set (if any).

Light Transport

Surface Cache estimates incoming hemispherical radiance per patch by shooting rays. The settings in this section control this process.

- **Multi Bounce**: When enabled, patches are allowed to query other patches when their rays hit geometry. This effectively approximates infinite bounces but only between patches which are allocated. When disabled, patches will not query other patches when their rays hit geometry, leaving you with single bounce lighting. Generally, you want this option to be enabled as it is relatively cheap.
- **Bounce Patch Allocation**: Surface Cache always spawn patches on surfaces visible by the camera. When enabling Bounce PatchAllocation, it will additionally allow patches to spawn patches wherever their rays hit. To avoid a chain reaction, only patches which are visible by the camera can spawn other patches. Enabling this makes multi-bounced light more accurate but it comes at a performance cost.
- **Sample Count**: The number of Monte Carlo samples to take per frame. Each sample involves 3 ray trace queries per active patch. Increasing this value will reduce temporal noise but it will also increase the runtime cost.

Patch Filtering

The patch lighting estimates are calculated via Monte Carlo integration (as is typical with ray tracing). This means that the results will be noisy, spatially and temporally. To mitigate noise Surface Cache filters across neighbouring patches and across frames. The settings in this section lets you control this filtering process.

- **Temporal Smoothing:** Controls the trade-off between lag and temporal noise. Large values will reduce temporal noise at the cost of introducing lag. Small values will reduce lag at the cost of causing more temporal noise.
    - 1.0: Ideal for slow-paced games.
    - 0.0: Best for highly dynamic scenes, ensuring responsiveness to fast-moving lights or objects.
- **Spatial Filtering Enabled**: Spatial Filtering is effectively blurring across nearby patches. The benefit of this is that this reduces spatial noise. The drawback is that fine details may be blurred out spatially. In some cases this may cause light leaking across planes with similar normals.
- **Spatial Filter Sample Count:** Defines how many neighboring lookups are used for the spatial filter. More samples means a higher quality output but the filter will be slower.
- **Spatial Filter Radius:** Defines the “patch space distance” used when searching for nearby patches. A value of 1 means that patches will look for other patches that are “one patch away” from them. Note that this distance is different across cascades (because patch sizes vary across cascades). Higher values will cause a wider blur. The radius should be sufficiently small to not cause overblur, yet large enough to hide noise. Usually 1 is a good starting point.
- **Temporal Post Filter:** Reduces high-speed noise, especially noise exacerbated by spatial filtering, by leveraging data from previous frames. This is almost always worth enabling. In certain cases you may want to stabilize the image in screen space instead.

Screen Filtering

The section configures how the discrete patch data is applied to the screen pixels. This is split into a low-resolution step which searches for nearby patches and a high-resolution step which performs upsampling into full resolution.

- **Lookup Sample Count:** The number of patch queries performed in the low-resolution step. More samples will reduce noise but it will also run slower. Usually, 5 is a good starting point.
- **Upsampling Kernel Size**: The radius of the upsampling filter in screen space. Larger values will cause more aggressive blur. The radius should be sufficiently small to not cause overblur, yet large enough to hide noise. Usually 2.5 is a good starting point.
- **Upsampling Sample Count:** The number of samples taken in the upsampling step. More samples will reduce noise but it will also run slower.

Blocky artifacts are often due to Lookup Sample Count being too low.

Fine grained noise is usually due to either too large Upsampling Kernel Size or too low Upsampling Sample Count.

Advanced

You'll find more advanced settings in this section.

- **Defrag Count**: The structure holding the patch data needs maintenance every frame. This settings controls how many defragmentation iterations to perform per frame.
    - For slow-paced games a small value like 2 is often enough.
    - Fast-paced games where large objects move or where the camera moves a lot usually require larger values.

Artifacts like this means that Surface Cache cannot allocate new patches. Increase Defrag Count to perform garbage collection more aggressively. Alternatively, you can reduce the total number of patches by reducing Resolution in the Volume section.

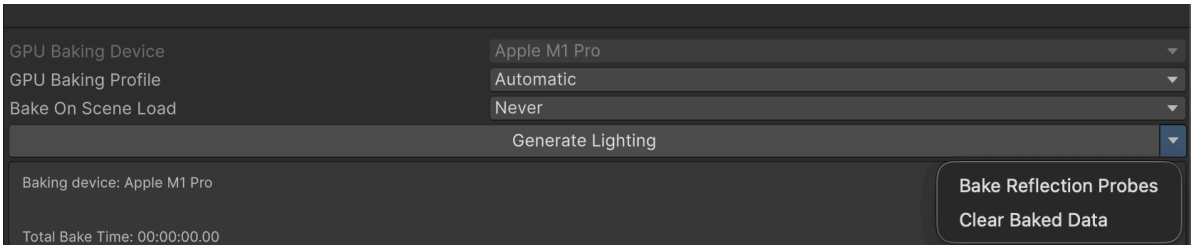
Comparison

In the table below you can see how Surface Cache compares to other GI methods in Unity.

	Lightmaps	**Light Probes**	**Adaptive Probe Volumes**	**Enlighten Realtime GI**	**Surface Cache GI**
Dynamic Geometry Contributing to GI	No.	No.	No.	No.	Yes.
Dynamic Geometry Receiving GI	No.	Yes, but large objects may show artifacts.	Yes.	Yes, from Light Probes. Large objects may show artifacts.	Yes.
Particle Support	No.	Yes.	Yes.	Yes.	No.
Dynamic Light Support	No.	No.	Some (Lighting Scenarios + Sky Occlusion).	Yes.	Yes.
CPU Cost	Very low.	Low (tetrahedron traversal).	Low.	Some.	Some.
GPU Cost	Very low.	Very low.	Some (indirection lookups, texture taps).	Low.	High.
RAM Usage	Low.	Low Some (tetrahedron structure)	Low.	High.	Minor.
VRAM Usage	Low.	Low.	Minor.	Low.	High (BVH, material pool, geometry pool).
Update Lag	N/A.	N/A.	None (using Lighting Scenarios).	Some.	Some.
Authoring Complexity	Medium/High (UV unwraps, configuration) .	Medium/High (manual probe placement).	Low/Medium (Auto placement, manual adjustment volumes, lighting scenarios).	Medium/High (UV unwraps, configuration).	Low.
Baking Time	High, editor only.	Low, editor only.	Low, editor only.	High, editor only.	None.
Visual Fidelity	Very High (solves on surface, high resolution, per pixel interpolation).	Low (solves in space, low resolution, per object interpolation).	Medium (solves in space, medium resolution, per pixel interpolation).	High for static lightmapped objects. Low for probe lit dynamic objects.	High (solves on surface, medium resolution, per pixel interpolation).
Platform Support	All.	All.	Compute shaders support required.	All.	PC, consoles. High-end handheld devices are WIP.
Leaking	Minor.	Some.	Some.	Some when using probes.	Minor.

Troubleshooting

- *Lighting is not realtime even though I have configured it to be realtime*: Unity considers all lights that have been baked as static. It does so regardless of whether you change Light Mode after a bake. Therefore, you may have to clear your baked lighting before a light is treated as realtime. You can clear baked lighting at the bottom of Window → Lighting.



- *Surface Cache doesn't work in player builds*: You must currently set Read/Write=True in the Mesh Importer of all meshes in a build. We expect to lift this limitation before final release.
- *Surface Cache is slow*: While Surface Cache GI was designed to be fast we cannot escape that any world-space GI solution will have some non-trivial cost. We recommend you try playing with the settings for example by reducing sample counts. We will be improving overall performance in the coming releases and will improve support for things like culling and LOD'ing.
- *Emissive Materials don't work well*: This is a known limitation. Please see the "Limitations" section. Note that large emissive triangles work better than small emissive triangles.
- *The output is noisy*: Try increasing the sample count in the "Screen Filtering" section of the Surface Cache GI Volume Override.
- *Interiors don't work well*: Many realtime GI systems struggle with interiors and/or dimly lit environments and Surface Cache GI is no exception. You can try increasing sample counts (this increases computational cost) and/or increasing Temporal Smoothing (this increases lag).
- *I see dark splotches when I move the camera*: This is most likely because Surface Cache's internal ring buffer has been exhausted. You can inspect its status by enabling Debug Mode (see Visualization Tools section above). If the bar is overwhelmingly Red this means that garbage collection cannot keep up. You can fix this by increasing Defrag Count (see Configuration section). If the bar is overwhelmingly yellow you may consider disabling Bounce Patch Allocation. Generally, you can try reducing the total number of patches by reducing Cascade Count or Resolution of the Surface Cache Volume (see Configuration section).

Limitations

Below is a list of limitations, some of which will be fixed before the final release.

- No support for animated materials. We may add support for this at a later point.
- **Static Batching** is not supported.

- **Alpha cut-outs** (like foliage) are not supported. All triangles are treated as opaque.
- HDRP and BiRP are not supported. Please see Render Pipelines strategy for 2026. That said, "Surface Cache Core" currently lives as a decoupled component in the SRP Core package allowing advanced users to integrate it into custom render pipelines.
- Surface Cache GI currently cannot combine with baked GI (e.g. lightmaps or probes). We do not expect to lift this limitation in time for 6.7 LTS but we may revisit this for a later release.
- There is currently no way to exclude objects from being present in Surface Cache's BVH. We expect to make this possible before the final release.
- Surface Cache may not yet work reliably on mobile phones. We expect to have some official support for mobile devices in time for 6.7 LTS but cannot make any guarantees at this point.
- For technical reasons, Surface Cache currently maintains copies of various data, such as mesh data and albedo texture data. This may contribute to high VRAM usage, especially in large scenes. We expect to be able to reduce this in the future.

Known Issues

Surface Cache is currently a work in progress and there are several known issues.

- Surface Cache does support emissive triangles but the results may be temporally noisy and/or slow to converge (depending on settings). We expect to improve support for emissive triangles in the future.
- For technical reasons, in order to make Surface Cache work in a build, you must set Read/Write=True in the Mesh Importer for all meshes that are part of the build. We are working on lifting this limitation.
- Surface Cache does not work well if you have both Scene View and Game View visible at the same time. We are working on a fix.
- Surface Cache does not work well with multiple active game cameras.
- Surface Cache covers a certain volume as defined in its settings. Outside of this volume Surface Cache currently provides no fallback GI; you will instead see black. We expect to address this before the final release, by adding a global fallback probe. For now, you may work around this issue by increasing the volume size.
- Flat surfaces that are perfectly aligned with the x, y, or z axes may cause flickering that looks like z-fighting. A work-around is to move these surfaces a tiny bit. We are working on a fix for this.
- Surface Cache currently cannot use hardware-accelerated raytracing on Metal and Vulkan (it will use the compute fallback instead). This is because Unity does not support hardware raytracing on these APIs. We are working on lifting this limitation.
- If Surface Cache is enabled during editor start-up and if using DX12, you may see no or reduced indirect light. This is due to a DX12 bug with metapass evaluation, which we expect to fix soon. The workaround is to disable, then enable Surface Cache Renderer Feature. You only have to do this once per editor session.
- Materials using Double-Sided GI are not yet supported. These will be supported in the future.

FAQ

Does Surface Cache require hardware ray-tracing?
No. Surface Cache is built on top of the UnifiedRayTracing library, and therefore it will use hardware raytracing when it is available and use a compute shader fallback when it is not.

Does Surface Cache work with Metal Ray Tracing or Vulkan Ray Tracing?
Surface Cache works with Metal and Vulkan but it cannot exploit hardware ray tracing on these APIs because Unity itself does not support this. Unity's Ray Tracing team is currently investigating the feasibility of adding support for Metal Ray Tracing and Vulkan Ray Tracing. If support is added, Surface Cache will automatically benefit from this.

What papers/techniques are Surface Cache based on?
Surface Cache is heavily inspired by Frostbite's GIBS, Kajiya renderer's irradiance cache, and Nvidia's SHARC. It is not a direct implementation of either of these. Instead, it takes aspects of each solution and combines them in a new way that is tailored for Unity's users.

What about specular indirect lighting?

Surface Cache GI is diffuse-only, but you may combine it with Reflection Probes and/or the upcoming Screen Space Reflections (aka SSR).